Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра  автоматики та управління в технічних системах

**Лабораторна робота № 13**

по дисципліні «Основи клієнтської розробки»

Тема: JavaScript. Браузер: документ(document) .

|  |  |
| --- | --- |
| Виконав:  студент групи ІА-94  Чумак Володимир Володимирович  Дата здачі \_\_\_\_  Захищено з балом \_\_\_ |  |

Київ 2020

**1. Тема**

JavaScript. Браузер: документ(document) .

**2. Завдання**

Використовувати тему, обрану в 4 роботі.

За допомогою об’єкта **document** змінити фон сторінки на 30 секунд.

За допомогою об’єкта **location** перенаправити браузер на іншу сторінку.

Викоритовуючи метод **getElementById**

Використати метод **querySelectorAll**

Використати наступні властивості DOM-вузла: **innerHTML**, **outerHTML**,

**nodeValue / data**, **textContent**

Внести зміни в документи/сторінку, використовуючи **document.write**,

**document.createElement(tag)**, **document.createTextNode(text)** та методи вставки

node.**append** (... nodes or strings), node.**prepend** (... nodes or strings), node.**after** (... nodes or strings), node.**replaceWith** (... nodes or strings), метод видалення вузлів node.**remove** ().

**3. Теоретичні відомості**

**DOM (Document Object Model)**

Document Object Model, скорочено DOM - об'єктна модель документа, яка

представляє весь вміст сторінки у вигляді об'єктів, які можна змінювати.

Об'єкт document - основна «вхідна точка». З його допомогою ми можемо

щось створювати або змінювати на сторінці.

**BOM (Browser Object Model)**

Об'єктна модель браузера (Browser Object Model, BOM) - це додаткові

об'єкти, що надаються браузером (оточенням), щоб працювати з усім, крім

документа.

Наприклад:

 Об'єкт navigator дає інформацію про сам браузер і операційну

систему. Серед безлічі його властивостей найвідомішими є: navigator.userAgent

- інформація про поточний браузер, і navigator.platform - інформація про

платформу (може допомогти в розумінні того, в якій ОС відкритий браузер -

Windows / Linux / Mac і так далі).

 Об'єкт location дозволяє отримати поточний URL і перенаправити

браузер за новою адресою.

**Навігація по DOM-елементів**

Всі операції з DOM починаються з об'єкта document. Це головна «точка

входу» в DOM. З нього ми можемо отримати доступ до будь-якого вузла.

Є одна тонкість: document.body може дорівнювати null

Не можна отримати доступ до елементу, якого ще не існує в момент

виконання скрипта.

Зокрема, якщо скрипт знаходиться в <head>, document.body в ньому

недоступний, тому що браузер його ще не прочитав.

Діти: childNodes, firstChild, lastChild

Тут і далі ми будемо використовувати два принципово різних терміни:

Дочірні вузли (або діти) - елементи, які є безпосередніми дітьми вузла.

Іншими словами, елементи, які лежать безпосередньо всередині даного.

Наприклад, <head> і <body> є дітьми елемента <html>.

Нащадки - все елементи, які лежать всередині даного, включаючи дітей,

їхніх дітей і т.д.

Колекція childNodes містить список всіх дітей, включаючи текстові вузли.

DOM-колекції

Як ми вже бачили, childNodes схожий на масив. Насправді це не масив, а

колекція – об’єкт перебору.

І є два важливих наслідки з цього:

1. Для перебору колекції ми можемо використовувати for..of:

for (let node of document.body.childNodes) {

alert(node); // покажет все узлы из коллекции

}

Це працює, тому що колекція є об'єктом перебору(є необхідний для цього

метод Symbol.iterator).

2. Методи масивів не працюватимуть, бо колекція - це не масив:

alert(document.body.childNodes.filter); // undefined (у коллекции нет

метода filter!)

Перший пункт - це добре для нас. Другий - буває незручний, але можна

пережити. Якщо нам хочеться використовувати саме методи масиву, то ми

можемо створити справжній масив з колекції, використовуючи Array.from:

alert( Array.from(document.body.childNodes).filter ); // сделали массив

DOM-колекції - тільки для читання

Майже всі DOM-колекції, за невеликим винятком, живі. Іншими словами,

вони відображають поточний стан DOM.

Якщо ми збережемо посилання на elem.childNodes і додамо / видалимо

вузли в DOM, то вони з'являться в збереженій колекції автоматично.

**4. Порядок виконання лабораторної роботи**

1) ознайомитися з теоретичними відомостями;

2) виконати завдання

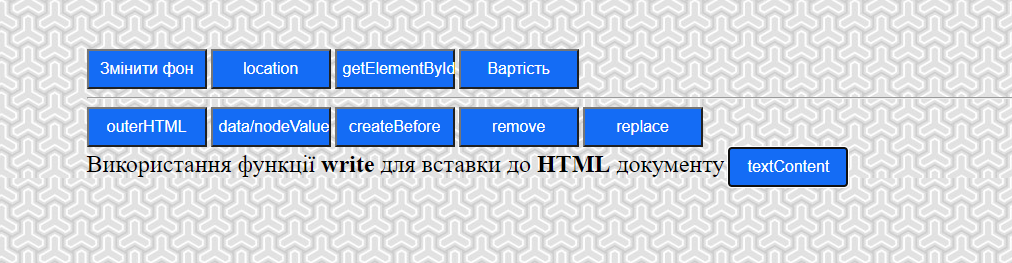
3) оформити звіт, який включатиме: титульний аркуш, завдання,

теоретичні відомості, результати і висновки по роботі;

4) продемонструвати результат на комп'ютері і захистити лабораторну

роботу.

**Результати виконання роботи**

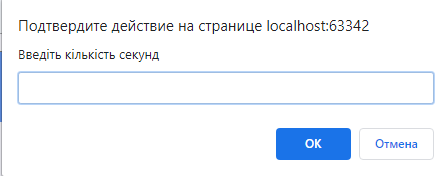


Частина файлу index.html

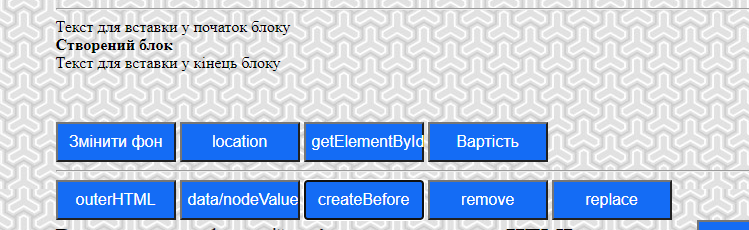


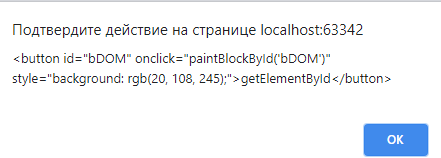
Частина файлу scripts.js

function *changeBack*() {  
 let sec = *prompt*('Введіть кількість секунд')  
 if (sec === '') {  
 *alert*('Поле порожнє')  
 } else if (sec < 0) {  
 *alert*('Введіть додатне число')  
 } else if (sec == null) {  
 *alert*('Ви відмінили дію')  
 } else {  
 document.body.style.backgroundImage = "url('images/laba13.jpg')"  
 *setTimeout*(() => document.body.style.backgroundImage = "", sec\*1000)  
 }  
}  
  
function *relocate*() {  
 location.href = "https://www.google.com.ua/?hl=ua"  
}  
  
function *paintBlockById*(id, color = "black") {  
 let block = document.getElementById(id)  
 block.style.background = color  
 *setTimeout*(() => block.style.background = "#146cf5", 1000)  
}  
  
function *changePrice*() {  
 let rowToChange = document.querySelectorAll('.t\_info > td')  
 let tdCount = document.querySelectorAll('.t\_info > td').length  
  
 for (let i = 1; i < tdCount; i += 10) {  
 rowToChange[i].onclick = function () {  
 let newPrice = *prompt*('Введіть ціну для заміни')  
 rowToChange[i].innerHTML = newPrice  
 }  
 }  
}  
  
function *createBefore*() {  
 let newBlock = document.createElement('div')  
 newBlock.innerHTML = "<br /><b>Створений блок</b><br /> "  
 let textBefore = document.createTextNode('Текст для вставки у початок блоку')  
 let textAfter = document.createTextNode('Текст для вставки у кінець блоку')  
 newBlock.append(textAfter)  
 newBlock.prepend(textBefore)  
 newBlock.className = 'newBlock'  
 document.querySelector(".dom").before(newBlock)  
}  
  
function *remove*() {  
 document.querySelector('.newBlock').remove()  
}  
  
function *replace*() {  
 let fInfoBox = document.querySelectorAll(".info\_box")[0]  
 let sInfoBox = document.querySelectorAll(".info\_box")[1]  
 fInfoBox.replaceWith(sInfoBox)  
}  
  
function *showTextContent*() {  
 let block = document.querySelectorAll(".info\_box")[0]  
 *alert*(block.textContent)  
}









**Висновки**

У цій лабораторній роботі я навчився використовувати об’єкти DOM та методи для управління цими об'єктами. Ознайомився із структурою DOM, способами отримати доступ до об’єктів та їх зміни.